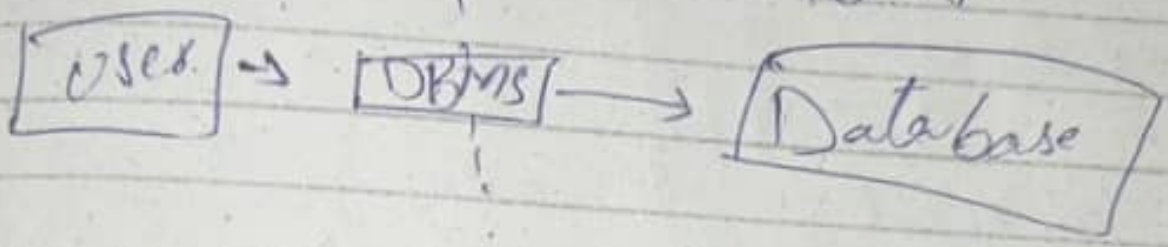
Conceptual Mapping

Database

جنيب

Database is collection of data in a format that can be easily accessed.
A Software application used to manage all database called DBMS.



Types of Databases

Relational	Non-relational
Data Store in tables	Data doesn't store in table
We use SQL to work with relational DBMS	

What is SQL?

SQL is a programming language used to interact with relational database.

It is used to perform "CRUD" operations

* Create
* Read
* Update
* Delete.

Creating Database

```
CREATE DATABASE db-name;
```

Delete database

```
DRop DATABASE db-name;
```

Use Database

```
USE db-name;
```

Creating Table

```
USE db-name;

CREATE TABLE table-name (
    column-name1 datatype constraint;
    column-name2 datatype constraint;
    column-name3 datatype constraint
);
```

id	name	age
1	Ali	20
2	Umar	18
3	Asad	21

```
CREATE TABLE Student(
id INT PRIMARY KEY,
name VARCHAR(50),
age INT NOT NULL
);
```

NOT NULL → (Should'nt be empty)
PRIMARY KEY → (id shouldn't be same / not dublicate)

Data insert into Table :-

```
INSERT INTO Student VALUES (1, "Ali", 20);
INSERT INTO Student VALUES (2, "Umar", 18);
INSERT INTO student VALUES (3, "Asad", 21);
```

For output

```
INSERT INTO Student (id, name, age)
VALUES (1, Ali, 20),
       (2, Umar, 18),
       (3, Asad, 21);
SELECT * FROM Student;
```

SQL Datatypes

They define the type of values that can be stored in a column.

Datatype	Description	Usage.
CHAR	String (0-255), can store characters of fixed length	CHAR(50)
VARCHAR	String (0-255), can store characters up to given length	VARCHAR(50)
BLOB	String (0-65535), Can store binary large obj	BLOB (1000)
INT	integer	INT
TINYINT	integer (-128 to 127).	TINYINT
BIGINT	integer	BIGINT

BIT	Can store x-bit values. x can range from 1 to 64	BIT(2)
FLOAT	Decimal values - with Precision to 23 digits	FLOAT
DOUBLE	Decimal values - with Precision to 24 to 53	DOUBLE
BOOLEAN	Boolean values 0 or 1	BOOLEAN
DATE	date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31	DATE
YEAR	year in 4 digits format from 1901 to 2155	YEAR

TINYINT UNSIGNED (0 to 255)

↓

(only for +ve integers)

TINYINT (-128 to 127)

↓

(Here the range will start from 0 if we use UNSIGNED datatype)

Types of SQL Commands

DDL (Data Definition Language): Create, alter, rename, truncate & drop)

DQL (Data Query language): Select

DML (Data Manipulation language): insert, update, delete

DCL (Data Control language): grant & revoke permission to users

TCL (Transaction Control language): Start transaction, commit, rollback

Database related Queries

CREATE DATABASE db_name;
CREATE DATABASE IF NOT EXISTS db_name;
↓
(create database if not exists

DROP DATABASE db_name; → (to delete db)
DROP DATABASE IF NOT EXISTS db_name;

SHOW DATABASES; (Show all databases in our server)
SHOW TABLE; (Show all tables of DB currently using)

DROP TABLE table_name; → (to delete table)

KEYS

Primary key

It is a column (or set of column) in a table that uniquely identifies each row (unique id).
There is only 1 PK & it should'nt be NOT null.
↓
(Primary key)

we can only declare one primary key in a table

Primary key

id	name	fee

Foreign key

A foreign key is a column (or set of column) in a table that refers to primary key of another table
There can be multiple foreign keys.
Foreign key have duplicate & null values.

Example

table 1 - student

id	name	city id
col	Ali	1
02	Umar	2
03	Asad	1

table 2 - City

id	city_name
1	Pune
2	delhi
3	Lahore

Foreign key for that table

Primary column of that table

CONSTRAINTS

SQL constraints are used to specify rules for data in a table.

NOT NULL: Columns can't have NULL value.
Syntax: col1 int NOT NULL;

UNIQUE: all values in column are different
Syntax: col2 int UNIQUE;

Example in Dry code

```
CREATE TABLE temp1 (
                    id INT UNIQUE
                  );

INSERT INTO temp1 VALUES (101);
INSERT INTO temp1 VALUES (101);
```

↳ It will give error because we assign UNIQUE constraint so we cannot duplicate value in this column.

PRIMARY KEY

makes a column unique & not null but used only for ~~more~~ one.

Syntax:

```
id INT PRIMARY KEY;
```

Alternative Syntax to declare PK

```
CREATE TABLE temp 1 (
    id INT,
    name VARCHAR (50),
    age INT,
    city VARCHAR(50),
    PRIMARY KEY (id) );
```

```
PRIMARY KEY(id, name)
```

↳ it is the combination of 2 col. is PK. One of them can be same but both can't be same

combination will be unique

Constraints

FOREIGN KEY

Prevent actions that would destroy links between tables.

```
CREATE TABLE temp(
    cust_id int,
FOREIGN KEY (cus_id) refrences customer(id)
```

DEFAULT

Set the default value of a column

Syntax.

```
Salary INT DEFAULT 25000
```

{ In this case, if someone leave the Sal
{ column empty, then default value will be set

Example.

```
CREATE TABLE empl (
    id INT,
    Salary INT DEFAULT 25000
    )

INSERT INTO empl (id) VALUES (101);
SELECT * FROM empl;
```

{ In this case, Salary will be shown 25000 even
{ we didn't set any salary bcz it is default constrain

CHECK

It can limit the values allowed in a column.

Example

```
CREATE TABLE city(
        id  INT PRIMARY KEY,
        city  VARCHAR(50),
        age  INT,
        CONSTRAINT age-chk CHECK (age>=18 AND city="Lahore")
        );
```

→ It is the name of constraint which is not necessary to write

Alternative method

```
CREATE TABLE newtab(
        id INT CHECK (age>=18)
        );
```

SELECT in Detail

use to select any data from database.

Basic Syntax

```
SELECT col1, col2 FROM t-name;
```

To Select All

```
SELECT * FROM t-name;
```

Column name ↓ table name ↓

SELECT DISTINCT city FROM Student;

{To select unique data e.g: only unique city names will be selected not repeated ones}

Where "clause" ← {It is an extra condition}

Example

SELECT * FROM student WHERE marks > 80;

{In that case, only those marks will be selected which are above 80}

Another Example

SELECT * FROM Student ~~WHERE~~ WHERE city = "Lahore";

SELECT * FROM student WHERE marks > 80 AND city = "Lahore";

{In this case, we apply two conditions}

Using Operators in WHERE

Arithmetic Operators: +, -, X, /, %

Example SELECT *
FROM Student
WHERE marks + 10 > 100

In that case the condition show marks of those stu whose total is above 100 aft adding 10 into their mar

1) Join Operations
2) Cartesian Product

Comparison Operators: =, !=, >, <, >=, <=

Example
```
SELECT *
FROM Student
WHERE marks >= 90;
```
In this case, only those marks selected which above or equal to 90

Logical Operator: AND, OR, NOT, IN, BETWEEN, All, LIKE, ANY

AND (To check for both conditions to be true)

```
SELECT * FROM Student WHERE marks>80 AND city="Lahore";
```

OR (To check one of the condition to be true)

```
SELECT * FROM Student WHERE marks>80 OR city="Lahore";
```

Between (Select for a given range)

```
SELECT * FROM Student WHERE marks BETWEEN 80 AND 90;
```

{It will show marks of student between 80 and 90, where 80 & 90 included}

IN (matches any value in the list)

```
SELECT * FROM Student WHERE city IN ("Lahore", "Karachi");
```

{It will show data of those student whom city values matches to given cities}

NOT (To negate given condition)

SELECT * FROM student WHERE city NOT IN ("Delhi", "Lahore", ...

{It will show data of those ~~cities~~ students whose city value doesn't match to the give cities }

Limit clause (Sets an upper limits on number of rows to be returned)

SELECT * FROM Students LIMIT 3 ;

{It will show data of just first 3 student}

ORDER BY clause (To sort in ascending (ASC) or descending (DESC) order)

SELECT * FROM student ORDER BY city ASC;

{it will show data according to ascending order of city}

SELECT * FROM student ORDER BY marks DESC;

{it will show marks in descending order)

SELECT * FROM Student ORDER BY marks DESC LIMIT 3;	It will show marks in descending order so we will get marks of top 3 students.

Aggregate Functions

Aggregate functions perform a calculation on a set of values and return a single value.

* COUNT()
* MAX()
* MIN()
* SUM()
* AVG()

Syntax:

```
SELECT max (marks)
FROM student;

SELECT COUNT(roll no)
FROM student;
```

Group By Clause

Groups rows that have the same values into summary rows.
It collects data from multiple records and groups the result by one or more column.

* Generally we use **group by** with some aggregation funtion.

Example: Count number of students in each city.

```
SELECT city, Count(name)
FROM student GROUP BY city;
```